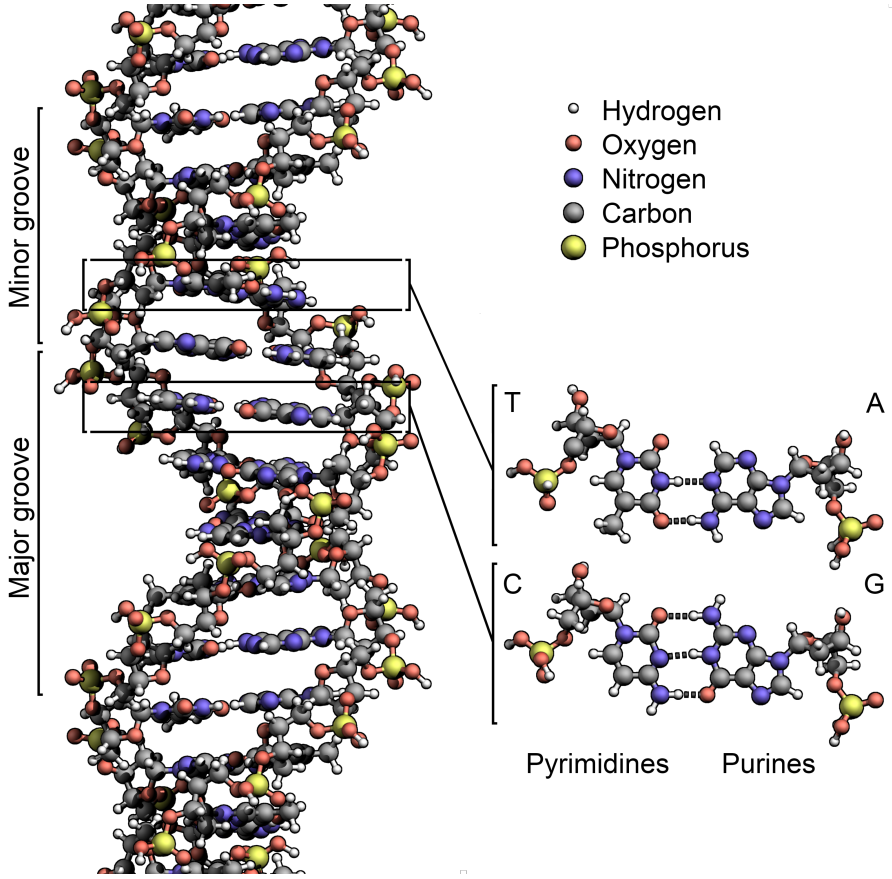
Objective C Programming

Project 1

DNA or Deoxyribonucleic acid is a nucleic acid containing the genetic instructions used in the development and functioning of all known living organisms. The four bases found in DNA are adenine (abbreviated A), cytosine (C), guanine (G) and thymine (T).



Any random part of DNA could look something like this: TTAGCTATAGCGCCTA.

1. Create a class called "Cell". It should inherit from NSObject. Your class should contain a variable called "DNA": an array of 100 characters to represent the DNA.

2. Write your own init method. It looks like this:

```
- (id) init {
    self = [super init];
    if (self) {
        // your initialization code goes here
    }

    return self;
}
```

So, we are overriding the default init method inherited from NSObject. But we still need to call it! So, the first line of our init method is calling the init of superclass and assigning the returned value to the current object (self). If it was successful then self has some value and your initialization code is executed.

Your initialization code should assign a value of A, T, G or C at random to each element in your array. So, after calling init on your object it should contain an array that looks something like this: [“T”, “G”, “T”, “A”, “C”, …] and be 100 elements long. It is up to you which data structure to use for single character (char, string, NSString, etc).

2. Create an instance method called “mutate”. It takes an integer **x** as a parameter and randomly changes **x** percent of DNA. For example, if x = 25, then 25% of elements are randomly changed to some value (A, T, C or G).

3. Create an instance method called “hammingDistance” that accepts another Cell as a parameter and returns an integer. This method should calculate the so-called Hamming Distance between two DNA strings: the number of positions in which corresponding elements are different. For example, strings:

A T **G** G C **A** T T T **A** G C
A T **A** G C **T** T T T **C** G C

have Hamming distance of 3, because there are three positions (marked bold) where corresponding values are not the same.

4. In your main function create an individual cell “alpha” and an array of 30. Mutate all the cells in the array.

5. Sort the array by hamming distance between “alpha” and the cell’s DNA in ascending order. So, the element with index 0 should be a cell with DNA that has the smallest hamming distance from “alpha” cell. The element with index 29 should be a cell with DNA that has the largest hamming distance from “alpha” cell.

6. Create a class called “SuperCell” which inherits from “Cell”. It should contain an additional variable called “DNA2” which looks the same as “DNA” of the parent class. Create “init” method that looks exactly like Cell’s init, but fills “DNA2” with random dna.

7. In your main function, take two cells from the array at a time and combine them into a supercell. So, take the DNA of the first cell (at index 0), take the DNA of the second cell (at index 1) and create a supercell with those two DNAs. Then, continue with index 2 and 3. You should end up with 15 SuperCells.